

An ejection-chain heuristic for the satellite downlink scheduling problem: A case study with RADARSAT-2*

Daniel Karapetyan^{†1}, Krishna T. Malladi^{‡1}, Snezana Mitrovic-Minic^{§2}, and
Abraham P. Punnen^{¶1}

¹*Department of Mathematics, Simon Fraser University Surrey, Central City, 250-13450
102nd AV, Surrey, British Columbia, V3T 0A3, Canada*

²*MDA Systems Ltd., 13800 Commerce Parkway, Richmond, British Columbia, V6V 2J3,
Canada*

Abstract

The synthetic aperture radar (SAR) technology enables satellites to efficiently acquire high quality images of the Earth surface. This generates a significant traffic from the satellite to the ground stations, and, thus, image downlinking may become a bottleneck in the efficiency of the whole system. In this paper we address the downlink scheduling problem for the Canada's Earth observing SAR satellite, RADARSAT-2. Being an applied problem, downlink scheduling is characterized with a number of constraints that make it difficult not only to optimize the schedule but even to produce a feasible solution. We propose a schedule generation procedure that lets us nicely incorporate all the constraints and then effectively optimize the schedule. Our computational experiments conducted on the real data show that the proposed algorithm is a significant improvement over the scheduling procedure currently in use.

Keywords: Satellite, Scheduling, Optimization, Heuristics.

1 Introduction

Efficient scheduling of image acquisition and image downlinking plays a vital role in satellite mission planning. These operations are often interlinked and solved using scheduling heuristics that take advantage of the flexibility allowed within such integrated systems. Most of the

*This research work was supported by an NSERC CRD grant awarded to Abraham P Punnen with the MDA Systems Ltd. as the collaborating industrial partner.

[†]daniel.karapetyan@gmail.com

[‡]kmalladi@sfu.ca

[§]snezanam@mdacorporation.com

[¶]apunnen@sfu.ca

literature on satellite mission planning (image acquisition and downlinking) is concentrated on satellites with optical devices for image acquisition [19, 21] or on modern satellites that use Synthetic Aperture Radar (SAR) technology [4, 5, 7, 9]. We study the mission planning operations of Canada’s Earth observing SAR satellite, RADARSAT-2.

Our problem deals with scheduling image downlinks optimally where an image acquisition schedule is given. The acquisition of RADARSAT-2 images is scheduled by a separate algorithm based on client requests using information about available time windows, and it is not considered in our study. The downlink scheduling process currently in operation for the RADARSAT-2 mission makes use of a greedy type algorithm followed by human intervention where necessary. The operators of RADARSAT-2 are facing increased demand for its imagery and are exploring various methods to improve the efficiency of the downlink scheduling algorithms. We developed efficient heuristics to handle the downlink scheduling which resulted in significant improvements. Experimental analysis using historical data established that for situations where downlink scheduling is oversubscribed a reduction in unsuccessful downlink requests of 20 to 30 percent is possible using our algorithm. It should be noted that the unsuccessful downlinks observed in the test data set are almost all for image requests that are considered low priority ‘background’ acquisitions. For test data sets from other seasons with lower order volumes our algorithm produced schedules with zero unsuccessful downlink requests whereas the existing model experienced a loss rate of 5 to 10 percent of low priority acquisitions. In all seasons, the total latency of downlinks is also reduced. These impressive experimental results were achieved using very large scale neighborhood (VLSN) search techniques [1, 2] along with special ejection chain schemes [14] to design the associated neighborhood structure.

The satellite image downlink scheduling problem (SIDSP) and its variations have been studied by many authors. Some of these works were concentrated on case studies for specific satellites [9] whereas other are more general purpose in nature [8, 10, 11, 12, 13, 18, 20, 21, 22]. Literature from machine scheduling [6, 17] and resource-constrained project scheduling [15] are also relevant in solving the SIDSP. However, each mission planning problem has its own restrictions and properties that can be exploited. Special care is needed to make sure that such restrictions are handled adequately which sometimes changes the inherent combinatorial structure of the problem significantly. Thus, investigating case studies of special SIDSPs are interesting and relevant as established in this study, although existing literature on the SIDSP considerably influenced our work.

The paper is organized as follows. In Section 2, we give a brief description of our SDISP and introduce various notations and definitions. Section 3 deals with our heuristic algorithms. Data analysis and generation of test instances are reported in Section 4 followed by computational results in Section 5 and concluding remarks in Section 6.

2 Problem Description

The SIDSP is defined by one satellite that orbits the Earth to acquire images and a set G of stationary ground stations that receive images from the satellite for further processing.

The image acquisition schedule is generated beforehand. Let R be a set of n image downlink requests that needs to be downlinked to a ground station during the planning horizon of 24 hours. For each request $r \in R$, a release time b_r , deadline e_r , downlink duration d_r , priority p_r and a ground station $g_r \in G$ are prescribed. The interval $[b_r, e_r]$ is called the *time window* of request r .

It may be noted that image downlinking needs to satisfy a variety of constraints and it may not be possible to downlink all the available images and, thus, rejection of requests is allowed. An image that is not downlinked before its deadline is considered ‘*unscheduled*’. The SIDSP deals with finding an image downlink schedule so that an appropriate utility function is maximized. The utility function considers the number of downlinks scheduled, their priority values, and the earliness of the downlink start time relative to its time window. The SIDSP is NP-hard since several NP-hard machine scheduling problems are special cases of it. Hereafter, the terminology *satellite* refers to the spacecraft RADARSAT-2, which is the focus of this study.

A downlink activity can be carried out only when the satellite is passing over the ground station coverage area, called *station visibility mask*, of the station. The satellite has two antennas that are used for downlinking and each ground station $g \in G$ has one or two channels that are used for receiving the downlinked image. If the ground station has two channels, two downlinks can happen in parallel (simultaneously) under certain circumstances which will be discussed later. In our case study, around 2/3 of the ground stations have one-channel capability and 1/3 of the stations have two-channel capability. Ground station channels work independently. Satellite antennas also work independently and are capable of downlinking to two different ground stations simultaneously.

Ground stations are also classified based on their transmission power. Let G_1 be the set of ground stations that are in *half-power setting* and G_2 be the set of ground stations that are in *full power setting*. Then $G = G_1 \cup G_2$ and $G_1 \cap G_2 = \emptyset$. When two images are downlinked one after another to stations with the same power setting, there must be a gap of δ seconds between the two downlinks. When two images are downlinked consecutively to two stations with different power settings, the required gap between the downlinks is $\Delta > \delta$ seconds. For full power ground stations, only one downlink is allowed at a time.

There are two ways of image downlinking: *pass-through mode* and *store-forward mode*. In pass-through mode, an image is downlinked as it is being acquired. This is possible only when the area to be imaged is within the visibility mask of the ground station to which it is downloaded. In store-forward mode, images taken earlier are stored in the satellite recorder and are downlinked as opportunity arises.

The visibility mask of a ground station $g \in G$ can be represented by a collection V_g of non-overlapping time intervals. We call V_g the *normal visibility mask*. Certain downlink requests require better reliability and in such cases we use finer components to define a visibility mask V_g^1 which we call *high reliability visibility mask*. Thus, for each $g \in G$, we have normal and high reliability visibility masks.

Let t_r be the scheduled downlink start time of request $r \in R$. If the request r is unscheduled, the value of t_r is undefined and this is denoted by $t_r = \emptyset$. The set $S = \{t_r : r \in R\}$

determines the solution to the SIDSP and it is called a *schedule*. Let $R(S) = \{r : r \in R \text{ and } t_r \neq \emptyset\}$ be the set of scheduled requests under S . Recall that the priority of a request $r \in R$ is p_r . Thus, a natural objective to consider is to maximize $\sum_{r \in R(S)} p_r$.

This objective measures the total priority level achieved by a downlink schedule. However, for the heuristics we considered, this objective produced poor solutions probably because the number of local maxima turned out to be very large. Since we are also interested in scheduling a request as early as possible, we have modified the objective function to be $f(S) = \sum_{r \in R(S)} p_r \left(1 - \alpha \cdot \frac{t_r - b_r}{e_r - d_r - b_r}\right)$, where $0 \leq \alpha \leq 1$ is a parameter reflecting the importance of scheduling each request downlink as early as possible. Note that when $\alpha = 0$, the objective function reduces to maximizing $\sum_{r \in R(S)} p_r$. Thus, our model for the SIDSP reduces to

$$\begin{aligned} \text{Maximize } f(S) &= \sum_{r \in R(S)} p_r \left(1 - \alpha \cdot \frac{t_r - b_r}{e_r - d_r - b_r}\right) \\ \text{subject to } S &\in \mathbb{F}, \end{aligned}$$

where \mathbb{F} is the collection of all schedules S satisfying the following constraints and assumptions:

- (1) A downlink cannot start earlier than the release time of the corresponding request and must be finished by its deadline. If a downlink cannot be done within this time window, the request is unscheduled.
- (2) If there are more than one request to downlink one image to different ground stations, either all or none must be scheduled.
- (3) There must be a gap of at least δ seconds between two consecutive downlinks under the same power setting.
- (4) The satellite antennas work either in half-power setting or full-power setting. The transition from one power setting to another requires $\Delta > \delta$ seconds. During the transition, none of the two antennas can work. Thus, if two consecutive requests are scheduled to two different ground stations with different power settings, then there has to be a time gap of at least Δ seconds between these downlinks.
- (5) If a satellite antenna is in the full-power setting, then only this one antenna can work and the other has to be idle. For both of the satellite antennas to be working independently, the antennas have to be in the half-power setting. An antenna in full power setting is used if and only if the corresponding ground station is in the full power setting.
- (6) We assume that the satellite is in the half-power setting at the beginning of the planning horizon and guarantee that it is in the half-power setting at the end. We also assume that downlinking (in half-power setting) can start right from the beginning of the planning

horizon, i.e., there was no downlink activity for at least δ seconds before the beginning of the planning horizon.

- (7) No downlink activity happens outside the planning horizon.
- (8) Once a downlink starts, it cannot be preempted, i.e., an image cannot be split.
- (9) A pass-through request is considered as *urgent* and, if possible, has to be scheduled at the earliest possible time in its time window.

Note that our objective function does not explicitly distinguish between urgent and non-urgent requests. However, this is taken care off by running our algorithm in two phases, where in the first phase all urgent requests are scheduled followed by scheduling of non-urgent requests in the second phase.

3 Development of the Algorithm

Our solution approach is based on a local search with an ejection chain neighborhood. A high-level description of the algorithm is given as follows:

Pre-processing: This step modifies the data to handle constraint (3).

Urgent requests processing: This step schedules all the urgent requests using the ejection chain algorithm.

Non-urgent requests processing: Starting with the partial schedules of urgent requests, we schedule non-urgent requests using the ejection chain algorithm, while maintaining the scheduled urgent requests intact.

3.1 Pre-processing

We first perform a pre-processing step which modifies the data to simplify the presentation of the algorithm and efficiently handle some of the constraints. One of the constraints of the SIDSP is to have a gap of at least δ units between two consecutive downlinks. To accommodate this, we add δ to d_r and e_r for all requests $r \in R$. Also, we increase by δ the upper bounds of each of the time intervals in normal and high reliability visibility masks for all $g \in G$. Thus, even if two downlinks are scheduled such that one is followed immediately by another, a gap of size δ between the downlinks is guaranteed.

Note that such an adjustment in the data prohibits some downlink activity at the end of the planning horizon which otherwise could have been scheduled. One can increase the end of the planning horizon by δ to take care of this. In that case, the conflicts between downlinks at the end of the previous planning horizon and the beginning of the current planning horizon can be taken care off by using appropriate perturbations and careful data handling.

Further, we replace Δ by $\Delta - \delta$ because the gap between downlinks to the stations when different power settings are used will always include the additional δ units of time we added. Hereafter, we assume that problem data is modified to reflect the changes discussed above and, hence, we can ignore explicit treatment of constraints (3).

3.2 Schedule Generator

This section describes the greedy algorithm we used to generate a schedule from a given ordered subset of requests R . Let us now discuss some terminology and notations to simplify the presentation of the algorithm and its validity proof.

Let V be a finite set where elements of V are non-intersecting intervals. We call sets of the type V *interval set*. Since the elements of V are non-intersecting intervals, V can also be viewed as an ordered set with the natural order produced by the position of these intervals on the real line. Thus, V is represented as $V = \{[\ell_1, u_1], [\ell_2, u_2], \dots, [\ell_v, u_v]\}$ where $\ell_1 < u_1 < \ell_2 < u_2 < \dots < \ell_v < u_v$ and $v = |V|$. We refer to the i th interval in V as V_i .

Consider an arbitrary interval $[a, b]$. Let us introduce sets of intervals I , L and R as follows:

$$\begin{aligned} B &= \{[\ell_j, u_j] \in V : \ell_j < b \text{ and } u_j > a\}, \\ C &= \{[\ell_j, a] : \exists j \text{ such that } [\ell_j, u_j] \in V \text{ and } \ell_j < a < u_j\} \text{ and} \\ D &= \{[b, u_j] : \exists j \text{ such that } [\ell_j, u_j] \in V \text{ and } \ell_j < b < u_j\}. \end{aligned}$$

Then subtracting the interval $[a, b]$ from V generates the set $V \ominus [a, b]$ of non-intersecting intervals given by

$$V \ominus [a, b] = (V \setminus B) \cup C \cup D.$$

The notation $V \ominus [a, b]$ is read as V minus interval $[a, b]$.

An interval $[a, b]$ said to be a *subinterval* of the intervals set V if there exists k such that $\ell_k \leq a < b \leq u_k$ and $[\ell_k, u_k] \in V$. This relationship is denoted by $[a, b] \in V$.

Let R^* be an ordered subset of R . Given R^* , we now present a *greedy algorithm* to schedule requests in R^* following the order prescribed for R^* . The algorithm maintains several indicator interval sets representing channel availability at ground stations and antenna availability on the satellite. After scheduling a request, the algorithm updates these indicator sets.

Let A be an interval set that represents the time intervals when both satellite antennas are available. Let H and F be interval sets indicating the time intervals when a half-power and full-power downlinks can happen, respectively. Let Q_g and Q_g^1 be interval sets indicating the availability of the ground station $g \in G$ in normal and high reliability visibility, respectively. Note that we need only one pair of indicator sets for a two channel ground station. Indeed, the number of channels in this case is not limiting since the number of simultaneous downlinks is constrained by the number of antennas. Hence, only the visibility of the ground station has to be considered.

The greedy algorithm works as follows:

1. Let $R^{**} \leftarrow R^*$.
2. Initialize the sets A , H and F to $\{[0, 24 \text{ hours}]\}$ which is the planning horizon. Initialize the set Q_g to the normal visibility mask of g and Q_g^1 to the high reliability visibility mask of g for every ground station $g \in G$. Note that for every $[\ell_j^1, u_j^1] \in Q_g^1$ there is an interval $[\ell_k, u_k] \in Q_g$ such that $\ell_k < \ell_j^1 < u_j^1 < u_k$.
3. Select the first request $r \in R^{**}$.
4. Suppose that r is downlinked to a half power station $g \in G_1$. Let $Q \leftarrow Q_g$ if r requires normal reliability and $Q \leftarrow Q_g^1$ otherwise. Choose the smallest $i \in \{1, 2, \dots, |H|\}$ and $j \in \{1, 2, \dots, |Q|\}$ such that $|H_i \cap Q_j \cap [b_r, e_r]| \geq d_r$. If no such i and j exist, the request r is unscheduled and $t_r \leftarrow \emptyset$. Otherwise $t_r \leftarrow x$, where $H_i \cap Q_j \cap [b_r, e_r] = [x, y]$.
 If $t_r \neq \emptyset$, the indicator sets are updated as follows. Let $X = [t_r, t_r + d_r]$. Apply $X \leftarrow X \ominus [\ell_i, u_i]$ for every $[\ell_i, u_i] \in A$. Now X contains all the intervals inside $[t_r, t_r + d_r]$ during which exactly one antenna was available. Apply $H \leftarrow H \ominus [\ell_i, u_i]$ for every $[\ell_i, u_i] \in X$. Also apply $A \leftarrow A \ominus [t_r, t_r + d_r]$ and $F \leftarrow F \ominus [t_r - \Delta, t_r + d_r + \Delta]$. Finally, if g is a one channel ground station, update $Q_g \leftarrow Q_g \ominus [t_r, t_r + d_r]$ and $Q_g^1 \leftarrow Q_g^1 \ominus [t_r, t_r + d_r]$.
5. Suppose that r is downlinked to a full power station $g \in G_2$. Let $Q \leftarrow Q_g$ if r requires normal reliability and $Q \leftarrow Q_g^1$ otherwise. Choose the smallest $i \in \{1, 2, \dots, |F|\}$ and $j \in \{1, 2, \dots, |Q|\}$ such that $|F_i \cap Q_j \cap [b_r, e_r]| \geq d_r$. If no such i and j exist, the request r is unscheduled and $t_r \leftarrow \emptyset$. Otherwise, $t_r \leftarrow x$, where $F_i \cap Q_j \cap [b_r, e_r] = [x, y]$.
 If $t_r \neq \emptyset$, the indicator sets are updated as follows: $F \leftarrow F \ominus [t_r, t_r + d_r]$ and $H \leftarrow H \ominus [t_r - \Delta, t_r + d_r + \Delta]$. Note that it is not necessary to update the indicator set A since no downlink can happen if neither H nor F is available.
6. Update R^{**} to $R^{**} \setminus \{r\}$ and, if $|R^{**}| > 0$, proceed to Step 3.
7. For every pair (r, u) of dual requests, if either $t_r = \emptyset$ and $t_u \neq \emptyset$ or $t_r \neq \emptyset$ and $t_u = \emptyset$, remove both r and u from R^* and proceed to Step 2.
8. Assign the downlinks to the particular ground station channels and satellite antennas.

Theorem 1. *The greedy algorithm terminates with a feasible schedule of requests in R^* in $O(n^3)$ time.*

Proof. Our updating scheme of the intervals sets A , H and F ensures that no antenna conflict arises and all the downlinks happen within the planning horizon. By subtracting $[t_r - \Delta, t_r + d_r + \Delta]$ from F for every half power downlink request r , we guarantee that no full power downlink will happen within Δ units from the downlink r . Similarly, no half power downlink can happen within Δ units from from a full power downlink r . Also, the updating of the indicator sets Q_g and Q_g^1 guarantees that no channel conflicts occurs and the downlinks obey visibility mask constraints. The pre-processing of data assures that there is

a gap of at least δ units between two consecutive downlinks. Finally, dual requests constraint is satisfied due to Step 7 of the algorithm. Thus, the schedule generated is feasible.

Let us now analyse the complexity of the algorithm. The primary operations in each iteration is: (1) to find the smallest i and j to satisfy certain condition and (2) update the indicator sets A , H , F , Q_g and Q_g^1 . Note that the size of the indicator sets Q_g and Q_g^1 for some $g \in G$ may increase but it is bounded by $O(n_g)$ where n_g is the number of requests to be scheduled to the station g . Similarly, the sizes of the indicator sets A , H and F are limited by $O(n)$. Thus, operation (1) for these sets can be performed in $O(n)$ time by simultaneous scanning of the sets. Operation (2) can also be performed in $O(n)$ time. Indeed, we only need to update a fixed number of indicator sets, and updating each of them takes $O(n)$ time (for a downlink to a half-power setting ground station, manipulating with X also needs only $O(n)$ time; note that each of the $H \leftarrow H \ominus [\ell_i, u_i]$ for $[\ell_i, u_i] \in X$ operations needs only $O(1)$ since all of these operations affect only one interval in H). The number of iterations is at most $O(n^2)$ and, thus, the algorithm complexity is $O(n^3)$. \square

3.3 Construction Heuristics

The greedy scheme discussed above can be used to obtain various construction heuristics for the SIDSP. Choose $R^* = R$ and depending on how we order elements of R^* we obtain different construction heuristics. We considered sorting the requests based on one or more of the following fields:

1. Priority p_r ;
2. Time window range $w_r^* = \lfloor e_r - b_r - d_r \rfloor$. Note that if we do not use rounding, time windows for all the requests could be different and, thus, sorting using a secondary criterion became somewhat superfluous.
3. Downlink duration d_r .

After extensive computational experiments using various combinations of the above fields to sort requests, it is determined that sorting elements of R^* with respect to p_r first, ties are broken by sorting with respect to w_r^* values and further ties, if any, are broken by d_r values provides the best performance. The resulting solution is used as the initial solution for our improvement heuristic that uses ejection chain neighborhoods. This Ejection Chain algorithm is described in the next section.

3.4 Ejection Chain Improvement Heuristic

We now show that we can find improved solutions using ejection chains. Note that the quality of the solution produced by the greedy algorithm depends on the ordering of elements in R^* . By repeating the algorithm with different orderings we get possibly different solutions and the overall best solution can be chosen. However, exhaustive searching over all such orderings is impractical. Our ejection-chain algorithm chooses ‘good’ ordering for R^* so that

the output generated is of high quality. Let \wp be the set of all permutations of elements of R . For any $\pi \in \wp$, let S_π be the schedule produced by the greedy algorithm by choosing $R^* = R$ and the ordering of elements of R is given by π . Thus, we concentrate on solutions $\{S_\pi : \pi \in \wp\}$ to design our ejection algorithm. In this sense, we consider \wp as the family of feasible solutions of the SIDSP. Thus, to solve the SIDSP, we essentially need to solve the *downlink requests permutation problem* (DRPP) given by:

$$\begin{aligned} &\text{Maximize } \phi(\pi) = f(S_\pi) \\ &\text{subject to } \pi \in \wp. \end{aligned}$$

Ejection chain based neighborhoods are commonly used in developing VLSN search algorithms [1, 2] for complex combinatorial optimization problems. For example, the well known Lin-Kernighan heuristic and its variations are ejection chain algorithms that are extremely efficient for the travelling salesman problem [3]. We now use the idea of the ejection chains [14] to develop a simple and effective heuristic to solve DRPP (and, hence, the SIDSP).

The data structure used in our ejection algorithm is a pair (π, h) , where π is a permutation of all the requests R and $h \in \{1, 2, \dots, n\}$ is a position in this permutation called *hole*. In order to calculate the objective $\phi(\pi, h)$ of the solution (π, h) , we choose $R^* = R$, order the element of R^* according to the permutation π and then remove the h th element from R^* . Then, by applying the greedy scheduling algorithm, we obtain a schedule and calculate its objective as defined above.

The basic move in our local search algorithm is swapping of an element with the ‘hole’, i.e., swapping $\pi(h)$ with $\pi(i)$ and updating h with i for some $i \neq h \in \{1, 2, \dots, n\}$. There are $n - 1$ options for this move, and we select the one that produces the best solution (π, h) . Every time we obtain a new solution (π, h) , we also evaluate the schedule S_π . If S_π is better than the best schedule known so far, we save it. For details of our ejection chain algorithm see Algorithms 1 and 2.

Note that, as it was mentioned above, this algorithm does not take care of the urgent requests. To make sure that the urgent requests are scheduled as early as possible, we start from running the ejection chain algorithm for the urgent requests only (see Section 3). This guarantees that they are scheduled as early as possible¹. Then we fix these requests and run the ejection chain algorithm for the regular requests.

4 Data analysis and generation of test instances

The algorithms developed in this paper were tested using real data. We used 13 *low density instances* collected between October 25 and December 13 of 2011, each containing approximately 100 requests per planning horizon. We also used 31 *high density instances* collected

¹In fact, we cannot guarantee that the urgent requests are scheduled as early as possible because we do not guarantee the optimality of our solutions. However, since the number of urgent requests is usually relatively small, corresponding subproblems are easy to solve, and, thus, we believe that the schedule containing only urgent requests is close to optimality.

Algorithm 1 Ejection Chain Algorithm.

Input: Requests R .

Input: Priorities p_r and time window durations w_r for each $r \in R$.

Input: Objective functions $\phi(\pi)$ and $\phi(\pi, h)$.

Create a permutation π of (R_1, R_2, \dots, R_n) such that $p_{\pi(i)} > p_{\pi(i+1)}$ (or $p_{\pi(i)} = p_{\pi(i+1)}$ and $\lfloor w_{\pi(i)} \rfloor < \lfloor w_{\pi(i+1)} \rfloor$, or $p_{\pi(i)} = p_{\pi(i+1)}$, $\lfloor w_{\pi(i)} \rfloor = \lfloor w_{\pi(i+1)} \rfloor$ and $d_{\pi(i)} < d_{\pi(i+1)}$) for every $i = 1, 2, \dots, n-1$.

Save π to a permutation σ .

Initialize the idle counter c with n . c keeps track of the number of non-improving iterations.

Initialize the hole position h with 1.

While there are no more than n consecutive non-improving iterations:

while $c > 0$ **do**

if $\text{improvement}(\pi, h, \sigma, 10) = 1$ **then**

 Reset the idle counter c with n .

 Save π to σ .

else

 Reduce the idle counter c by one.

end if

 Go to the next hole position: if $h = n$ set $h \leftarrow 1$ otherwise $h \leftarrow h + 1$.

end while

return σ .

in August 2011, each containing approximately 300 requests. There are about 10 ground stations involved in each instance and these ground stations are typically visible during the planning horizon from 4 to 10 times.

For the low density data, two solutions were given to us: *machine-scheduled* and *human-rescheduled*. The machine scheduled solutions are generated by the algorithm currently in use for mission planning. Such a solution may be infeasible and the human-rescheduled solution modifies the machine scheduled solution to remove violations and to satisfy some additional considerations known to the operators at that time. Such a data (and solution) may be imprecise. For example, our model uses crisp visibility mask boundaries but a human operator may use his/her judgement to schedule a job even if a job goes beyond the prescribed visibility mask by very small amounts but such a solution will be infeasible as per our model.

The high density instances were given to us as a single file containing data for the month of August 2011. In particular, we were provided with a table of all the requests and the schedule generated by human intervention. Since we were not provided any explicit instances for each planning horizon of 24 hours, we used the following rules to construct the instances. Let T_i be the planning horizon of the i th instance, $i = 1, 2, \dots, 31$.

1. If a request r is present in the provided schedule, we assign it to the i th instance, where $t_r \in T_i$.

Algorithm 2 Ejection Chain Algorithm: $improvement(\pi, h, \sigma, d)$.

Input: Permutation π with and a hole position h .

Input: The best solution σ found so far.

Input: The remaining search depth d , i.e., potentially the longest ejection chain

```

if  $d = 0$  then
    return 0.
end if
Save the objective value of the partial solution  $\phi(\pi, h)$  to  $\phi_p$ .
Initialize  $j$  with  $\emptyset$ .
for  $i \leftarrow 1, 2, \dots, h-1, h+1, \dots, n$  do
    Swap  $\pi(h)$  with  $\pi(i)$ .
    if  $\phi(\pi) > \phi(\sigma)$  then
        Save the solution  $\pi$  to  $\sigma$ .
        return 1.
    end if
    if  $\phi(\pi, i) > \phi_p$  then
        Update  $j$  with  $i$ .
        Update  $\phi_p$  with  $\phi(\pi, i)$ .
    end if
    Swap  $\pi(h)$  with  $\pi(i)$ .
end for
if  $j \neq \emptyset$  then
    Apply the modification by swapping  $\pi(h)$  and  $\pi(j)$ .
    if  $improvement(\pi, j, \sigma, d-1) = 1$  then
        return 1.
    end if
end if
return 0.

```

2. If an urgent request r is withdrawn in the provided schedule, we assign it to the first instance where it can be downlinked.
3. If a regular request r is withdrawn in the provided schedule, we assign it to the i th instance, where i is chosen such that the length of the interval $T_j \cap [b_r, e_r - d_r]$ is maximized over $i \in \{1, 2, \dots, 31\}$ (if there are several such i s, we chose the smallest one).

Several of the 31 instances were excluded from our test bed because of a significant amount of violations found in the provided solution. These are instances corresponding to Aug 7, Aug 8 and Aug 31.

5 Computational Experiments

The ejection chain algorithm is coded in C++ and tested on a PC with Intel i7-2600 CPU. The low and high density instances as discussed above were used as our test bed. We compared our results with the solutions provided by our industrial partner. The summary of our experimental results on low density instances is given in Table 1.

Instance	n	Urg.	To the best, %			Urg. unsch.			Total unsch.			Avg. urg. del., s			Avg. reg. del., s		
			M-S	H-R	EC	M-S	H-R	EC	M-S	H-R	EC	M-S	H-R	EC	M-S	H-R	EC
Oct 25	110	31	1.7	1.7	0.0	0	0	0	1	1	0	0.4	0.3	0.0	712	762	135
Oct 28	108	34	2.1	1.2	0.0	0	0	0	3	1	0	0.4	0.5	0.0	445	1001	229
Nov 1	115	29	1.9	1.9	0.0	0	0	0	1	1	0	0.7	0.7	0.0	881	881	94
Nov 15	105	25	2.5	2.5	0.0	0	0	0	1	1	0	0.7	0.7	0.0	1613	1613	513
Nov 17	104	32	1.5	2.0	0.0	0	0	0	1	2	0	6.9	0.4	0.0	711	701	184
Nov 22	105	33	1.1	1.1	0.0	0	0	0	1	1	0	0.3	1.3	0.0	201	202	119
Nov 23	90	30	1.2	1.1	0.0	0	0	0	1	1	0	0.4	3.1	0.0	371	379	79
Nov 29	108	31	1.0	1.0	0.0	0	0	0	1	1	0	4.5	4.5	0.0	234	234	102
Dec 1	110	28	1.9	1.6	0.0	0	0	0	2	1	0	5.7	11.3	0.0	536	975	237
Dec 2	134	27	0.3	0.3	0.0	0	0	0	0	0	0	0.4	0.4	0.0	564	564	65
Dec 7	108	36	1.1	1.1	0.0	0	0	0	1	1	0	1.4	0.5	0.0	573	575	151
Dec 13	94	24	2.1	2.1	0.0	0	0	0	2	2	0	0.2	0.2	0.0	566	566	40
Average	107.6	30.0	1.5	1.5	0.0	0.0	0.0	0.0	1.3	1.1	0.0	1.8	2.0	0.0	617	704	162

Table 1: Comparison of the given machine-scheduled solutions (M-S), human-rescheduled solutions (H-R) and the results of our ejection chain algorithm (EC) for the low density instances.

Each row of the table corresponds to one problem instance. For each of the algorithms and for each of the instances we provided how much worse is the solution comparing to the best known solution for this instance (‘To the best, %’), the number of unscheduled urgent requests (‘Urg. unsch.’), the total number of unscheduled requests (‘Total unsch.’), the average delay of the urgent downlinks (‘Avg. urg. del., s’) and the average delay of the regular downlinks (‘Avg. reg. del., s’). Since the machine-scheduled solutions are sometimes infeasible, we consider a downlink to be scheduled if it falls into the planning horizon and is not present in the violations table provided with the machine-scheduled solution. The delay of a request r is measured as $t_r - \tau_r$, where τ_r is the earliest time when r can be downlinked.

Note that the comparison of our schedules to the human-rescheduled solutions may be incorrect. Indeed, we may not know some of the constraints that are taken into consideration by the operator. Thus, we focus on the comparison to the machine-scheduled solutions.

The computational results show that the solutions produced by our algorithm dominate the provided machine-scheduled solutions for all of the instances. In particular, our algorithm always schedules all the downlinks while usually there were several unscheduled requests in the given solutions. Our algorithm never delays the urgent requests while it sometimes happens in the given machine-scheduled solutions, i.e., there are large delays (around 3 minutes) for the Nov 17 and Nov 29 instances. In each case, the delay is mainly caused by only one downlink that the algorithm failed to schedule to the beginning of the time window.

Note that, since our algorithm schedules all the requests for every low density instance,

the optimality of schedules does not depend on the value of the constant α (see Section 2).

The running time of our algorithm is always very reasonable and never exceeds 10 seconds in our experiments.

Computational results for high density instances are given in Table 2.

Instance	n	Urg.	To the best, %		Urg. unsch.		Total unsch.		Avg. urg. del., s		Avg. reg. del., s	
			H-R	EC	H-R	EC	H-R	EC	H-R	EC	H-R	EC
Aug 1	211	34	13.5	0.0	0	0	53	19	0.3	0.0	2013	457
Aug 2	301	35	20.5	0.0	0	0	123	67	0.6	0.0	3788	2725
Aug 3	324	42	20.7	0.0	0	0	120	48	0.5	0.0	2329	2248
Aug 4	294	41	21.1	0.0	2	0	116	52	0.5	0.0	2025	1248
Aug 5	260	36	13.3	0.0	0	0	85	49	0.5	0.0	2621	1021
Aug 6	356	46	20.0	0.0	2	1	130	61	0.4	0.0	2555	1563
Aug 9	259	38	18.7	0.0	3	0	87	30	92.4	0.0	2448	1867
Aug 10	304	54	20.9	0.0	1	0	98	33	0.4	0.0	2230	1338
Aug 11	278	39	19.4	0.0	3	0	90	36	0.5	0.5	2463	904
Aug 12	230	25	16.8	0.0	0	0	80	28	0.5	0.0	2880	2096
Aug 13	324	41	16.7	0.0	1	0	101	38	0.4	0.0	2045	868
Aug 14	285	34	18.8	0.0	4	3	104	54	0.4	0.0	1558	818
Aug 15	259	30	18.6	0.0	0	0	93	43	0.4	0.0	1692	1303
Aug 16	299	39	29.5	0.0	4	0	139	54	0.4	0.0	1374	1849
Aug 17	303	30	23.0	0.0	0	0	128	65	0.5	0.0	2664	1193
Aug 18	275	33	20.6	0.0	2	1	115	54	0.4	0.0	2557	1819
Aug 19	254	30	18.8	0.0	0	0	93	34	0.4	0.0	2210	1817
Aug 20	295	39	19.4	0.0	0	0	96	35	0.4	0.0	2768	1124
Aug 21	272	24	18.7	0.0	0	0	100	41	0.4	0.0	2531	1479
Aug 22	253	36	18.0	0.0	0	0	91	45	0.5	0.0	1737	1486
Aug 23	307	44	18.6	0.0	0	0	111	45	0.3	0.0	2450	1421
Aug 24	300	40	22.9	0.0	1	0	122	56	0.4	0.0	1465	1146
Aug 25	269	40	36.2	0.0	11	0	112	48	2208.0	0.0	6051	1664
Aug 26	301	37	19.2	0.0	0	0	116	44	0.5	0.0	1955	2152
Aug 27	282	40	18.5	0.0	0	0	89	33	0.4	0.0	2720	921
Aug 28	282	34	21.9	0.0	5	0	108	41	0.5	8.0	2765	1014
Aug 29	267	36	20.5	0.0	3	1	95	46	0.4	0.0	2991	1393
Aug 30	315	44	21.0	0.0	1	0	116	46	0.3	0.0	2081	1582
Average	284.3	37.2	20.2	0.0	1.5	0.2	104.0	44.5	82.6	0.3	2463	1447

Table 2: Comparison of the human-rescheduled (H-R) solutions with the results of our ejection chain algorithm (EC) for the high density instances.

Unlike it was for the low density instances, our ejection chain algorithm is not able to schedule all the requests in the high density instances. However, it schedules 60 requests more, on average, than in the given machine-scheduled solutions. It also schedules more urgent requests and provides a smaller average downlink delay. The running time of the algorithm is below 4 minutes on average and it never exceeds 10 minutes.

Another experiment that we conducted was to schedule as many requests from the high density instances as possible. For this purpose we defined the instances as follows. The **Aug 1** instance includes all the requests that can be downlinked on August 1. The **Aug 2** instance includes all the requests that can be downlinked on August 2 except the requests scheduled for **Aug 1**. The **Aug 3** instance includes all the requests that can be downlinked on August 3 except the requests scheduled for **Aug 1** and **Aug 2**, etc.

We call such instances *rolling*. Note that rolling instances may be different for different algorithms and, thus, comparison of the performance of the algorithms for every particular instance is meaningless.

The results of our experiment with the rolling instances are reported in Table 3.

Algorithm	Objective	Scheduled	Unscheduled	Avg. delay, sec	Running time, hours
Human-rescheduled	219.1	5056	3428	9486.6	—
Construction	293.8	6513	1971	12286.6	0.0
Ejection chain	304.1	7084	1400	8995.6	3.9

Table 3: Comparison of the given human-rescheduled solution with the results of our construction heuristic and our ejection chain algorithm for the rolling instances.

It takes around 4 hours for our ejection chain algorithm to schedule 7086 out of 8484 requests for the entire month of August, compared to 5056 requests scheduled in the human-rescheduled solution. Note that our algorithm also improved the average delay and, hence, the quality of the solution.

Our construction heuristic proceeds in almost no time and schedules more requests than in the given human-rescheduled solutions. However, the average delay in this solution is higher.

Although such an experiment may not reflect the real procedure of scheduling downlink requests, it shows the potential for improving downlink schedules.

5.1 The parameter α

Recall that the objective function for the SIDSP uses a parameter α that represents the relative preference of the downlink delays over unscheduled requests. Our ejection chain algorithm was able to schedule all the requests for the low density instances and, hence, the solutions obtained for these instances were almost independent of the value of α . In contrast, for the high density instances, careful selection of the value of α was crucial in achieving high quality solutions. In Figure 1 we show how the performance of our ejection chain algorithm depends on the value of α for the high density instances.

The tendency is that as α increases, the delay in scheduling a request decreases and the number of unscheduled requests increases. Hence, α can be used as a parameter to adjust the trade-off between the number of scheduled requests and the average delay.

Note that the performance of the algorithm is poor for $\alpha = 0$. This is because the objective in this case does not depend on the delays making the search landscape more discrete and, thus, complicated for optimization. It is also worth mentioning that the difference in the solutions obtained for different values of α is relatively small.

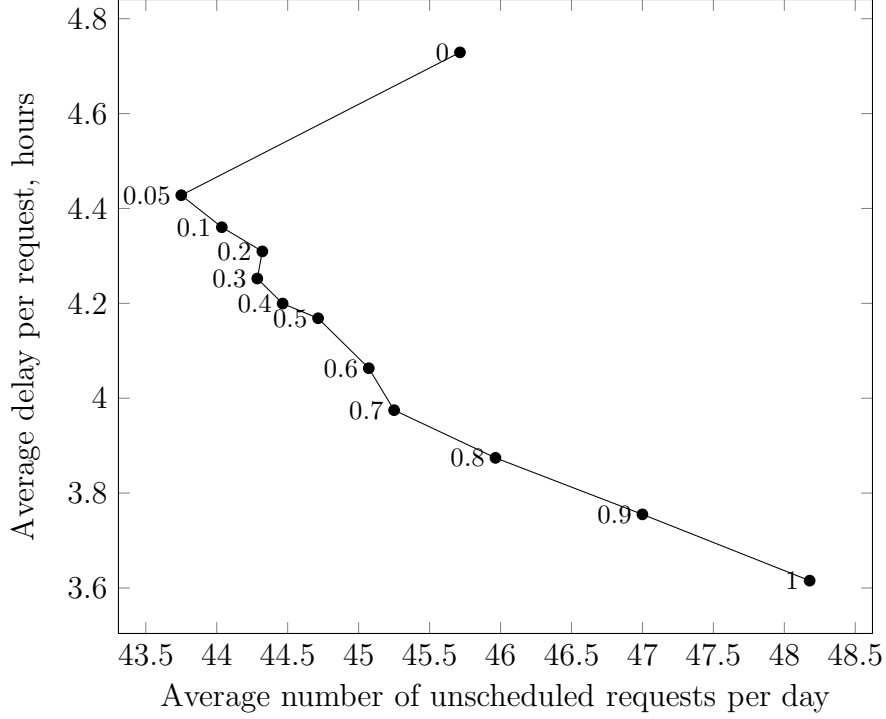


Figure 1: How the value of α influences the performance of our ejection chain algorithm. The experiments were conducted for the high density instances.

6 Conclusions

In this project, we formalized the satellite downlink scheduling problem and proposed an efficient heuristic solution approach. Computational experiments show that our algorithm clearly outperforms the currently implemented algorithm, and the running time of our algorithm is practically negligible to reasonable.

It is worth noting that we understand that the proposed problem definition may not be precise and some additional constraints may be found out later. Our approach, however, is very flexible and we believe that it can easily incorporate potential additional constraints.

References

- [1] R.K. Ahuja, O. Ergun, J.B. Orlin, A.P. Punnen, A survey of very large scale neighborhood search techniques, *Discrete Applied Mathematics* 123 (2002) 75-102.
- [2] R.K. Ahuja, O. Ergun, J.B. Orlin, and A.P. Punnen, Very Large Scale Neighborhood Search: Theory, Algorithms and Applications, *Approximation Algorithms and Metaheuristics*, T. Gonzalez (ed), CRC Press, 2007.

- [3] D. Gamboa, C. Osterman, C. Rego and F. Glover, An Experimental Evaluation of Ejection Chain Algorithms for the Traveling Salesman Problem. Technical report, School of Business Administration, University of Mississippi, 2006.
- [4] L. Barbulescu, J.P. Watson, L.D. Whitley, and A.E. Howe, Scheduling space-ground communications for the Air Force Satellite Control Network, *Journal of Scheduling*, 7 (2004) 7-34.
- [5] L. Barbulescu, L. D. Whitley, and A. E. Howe, Leap before you look: An effective strategy in an oversubscribed scheduling problem, 2004.
- [6] J. F. Bard and S. Rojanasoonthon, A branch and bound algorithm for parallel machine scheduling with time windows and job priorities, *Naval Research Logistics* 53 (2006) 24-44.
- [7] G. Beaumet, G. Verfaillie, M.-C. Charneau, Feasibility of autonomous decision making on board an agile earth-observing satellite, *Computational Intelligence*, 27 (2011) 123-139.
- [8] N. Bianchessi, J.-F. Cordeau, J. Desrosiers, G. Laporte, and V. Raymond, A heuristic for the multi-satellite, multi-orbit and multi-user management of Earth observation satellites, *European Journal of Operational Research* 177 (2007) 750-762.
- [9] N. Bianchessi and G. Righini, Planning and scheduling algorithms for the COSMO-SkyMed constellation, *Aerospace Science and Technology* 12 (2008) 535-544.
- [10] A. Globus, J. Crawford, J. Lohn, and A. Pryor, A comparison of techniques for scheduling earth observing satellites, In *Proceedings of the sixteenth innovative applications of Artificial Intelligence Conference (IAAA-04)*, San Jose, California, 2004.
- [11] T. Benoist and B. Rottembourg, Upper bounds for revenue maximization in a satellite scheduling problem, *4OR* 2 (2004) 235-249.
- [12] J.-F. Cordeau and G. Laporte, Maximizing the value of an earth observation satellite orbit. *Journal of the Operational Research Society*, 56 (2005) 962-968.
- [13] V. Gabrel and C. Murat, Mathematical programming for Earth observation satellite mission planning. In *Operations research in space and air*, Boston: Kluwer Academic (Chap. 7) 2003.
- [14] F. Glover, Ejection Chains, Reference Structures and Alternating Path Methods for Traveling Salesman Problems, *Discrete Applied Mathematics* 65 (1996) 223-253.
- [15] S. Hartmann, D. Briskorn, A survey of variants and extensions of the resource-constrained project scheduling problem. *European Journal of Operational Research* 207 (2010), 1-14.

- [16] F. Marinelli, F. Rossi, S. Nocella and S. Smriglio, A Lagrangian heuristic for satellite range scheduling with resource constraints, *Optimization Online*, 2006.
- [17] S. Rojanasoonthon and J. Bard, A GRASP for parallel machine scheduling with time windows, *INFORMS Journal on Computing* 17 (2005) 32-51.
- [18] B. Sun, W. Wang, X. Xie, and Q. Qin, Satellite mission scheduling based on genetic algorithm, *Kybernetes* 39 (2010) 1255-1261.
- [19] M. Vasquez and J.-K. Hao, A logic-constrained knapsack formulation and a tabu algorithm for the daily photograph scheduling of an earth observation satellite. *Computational Optimization and Applications* 7 (2003) 87-103.
- [20] G. Verfaillie, C. Pralet, M. Lemaitre, How to model planning and scheduling problems using constraint networks on timelines. *Knowledge Engineering Review*, 25 (2010) 319-336.
- [21] P. Wang, G. Reinelt, P. Gao, Y. Tan, A model, a heuristic and a decision support system to solve the scheduling problem of an earth observing satellite constellation, *Computers & Industrial Engineering* 61 (2011) 322-335.
- [22] N. Zufferey, P. Amstutz, and P. Giaccari, Graph colouring approaches for a satellite range scheduling problem. *Journal of Scheduling* 11 (2008) 263-277.